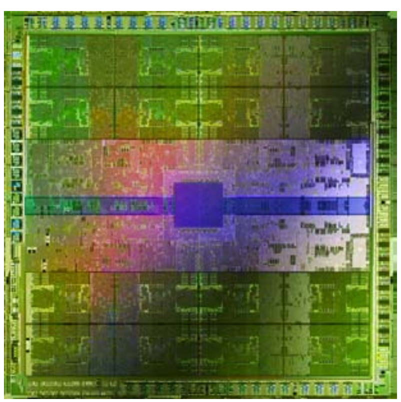
Using GPUs for Weather and Climate Models

Mark Govett
ESRL/GSD

GPU / Multi-core Technology

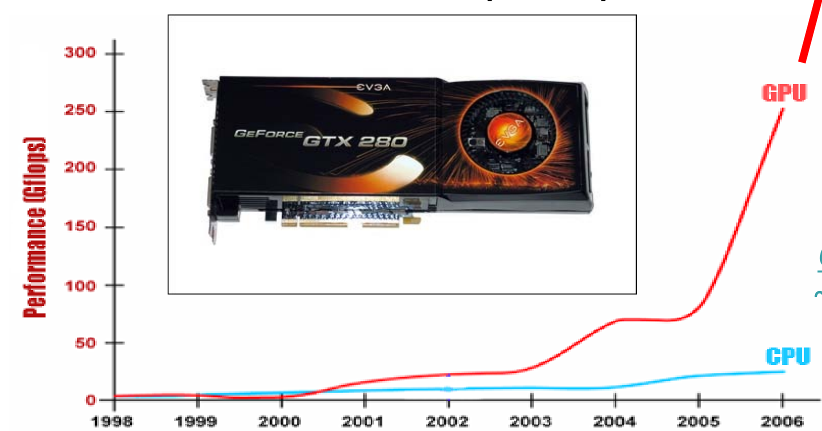
- NVIDIA: Fermi chip first to support HPC
  - Formed partnerships with Cray, IBM on HPC systems
  - #2 system on TOP500 (Fermi, China)
- AMD/ATI: Primarily graphics currently
  - #7 system on TOP500 (AMD-Radeon, China)
  - Fusion chip in 2011 (5 TeraFlops)
- Intel: Knights Ferry (2011), 32-64 cores

NVIDIA: Fermi (2010)



✧ 1.2 TeraFlops
✧ 8x increase in double precision
✧ 2x increase in memory bandwidth
✧ Error correcting memory

NVIDIA: Tesla (2008)



GPU: 2008
933Gflops
150W

CPU:2008
~45 Gflops
160W

CPU – GPU Comparison

CHIP TYPE	CPU Nahalem	GPU NVIDIA Tesla	GPU NVIDIA Fermi
Cores	4	**240**	**480**
Parallelism	Medium Grain	Fine Grain	Fine Grain
Performance Single Precision Double Precision	 47 Gflops 23 GFlops	 **933 GFlops** **60 GFlops**	 **1040 GFlops** **500 GFlops**
Power Consumption	130W	150W	220W
Memory	24-48 GBytes	**1-2 GBytes**	**3-6 GBytes**

Next Generation Weather Models

- Models being designed for global cloud resolving scales (3-4km)
- Requires PetaFlop Computers

<u>DOE Jaguar System</u>

- 2.3 PetaFlops
- 250,000 CPUs
- 284 cabinets
- 7-10 MW power
- ~ $50-100 million
- **Reliability in hours**

<u>GPU System</u>

- 1.0 PetaFlop
- 1000 NVIDIA GPUs
- 10 cabinets
- 0.5 MW power
- ~ $5-10 million
- **Reliability in weeks**

- Large CPU systems (~100 thousand cores) are unrealistic for operational weather forecasting
  - Power, cooling, reliability, cost
  - Application scaling

Valmont
Power Plant
~200 MegaWatts
Boulder, CO

Programming GPUs

- Languages
  - CUDA-C: available from NVIDIA
  - OpenCL: industry standard (NVIDIA, AMD, Apple, etc)
  - Fortran: PGI, CAPS, F2C-ACC compilers
- Fine grain (loop level) parallelism
  - Needed to keep 480+ cores busy
  - Code modifications, restructuring may be necessary to get good performance

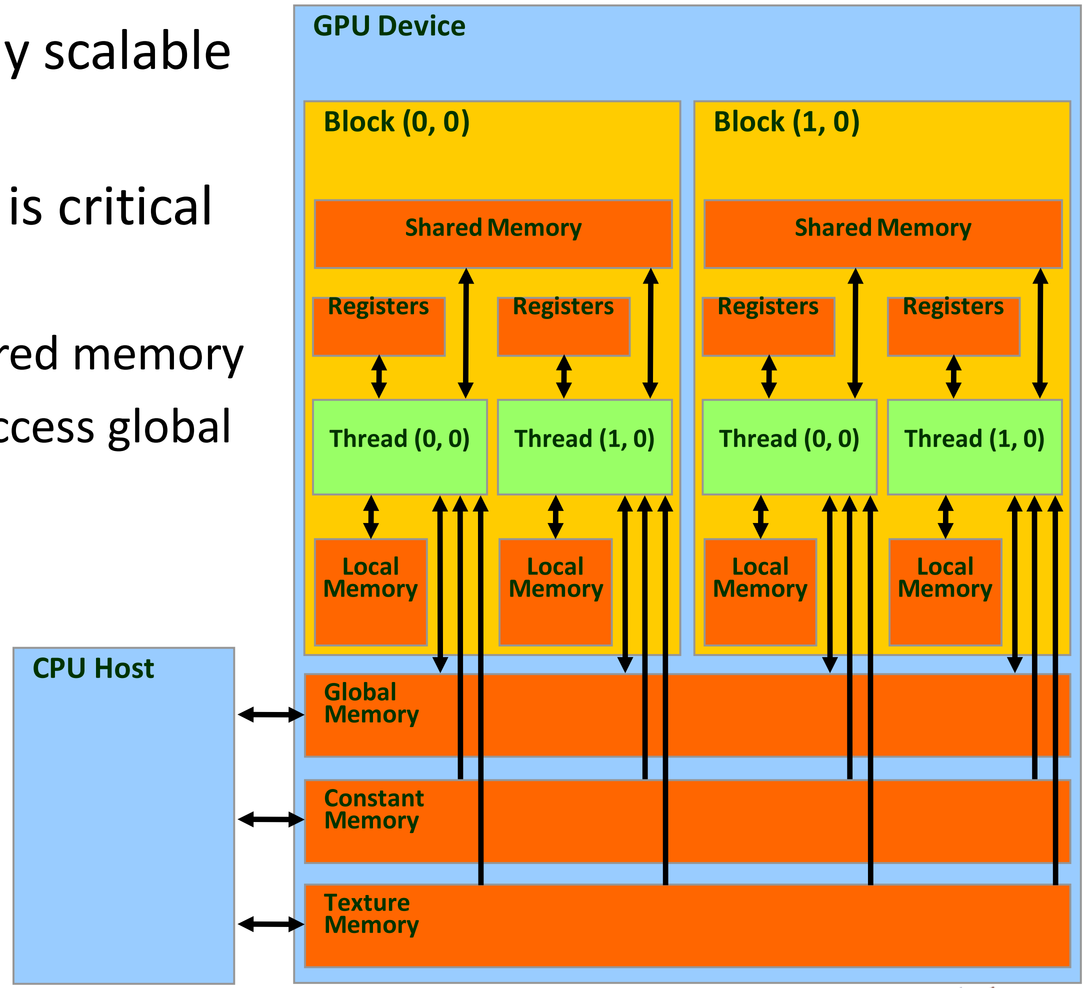
Application Performance

- 100x is possible on highly scalable codes
- Efficient use of memory is critical to good performance
  - 1-2 cycles to access shared memory
  - Hundreds of cycles to access global memory

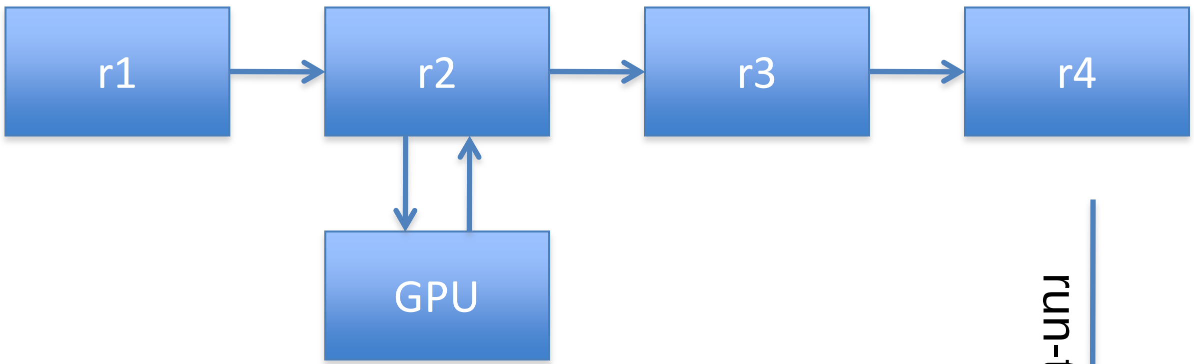
<u>Tesla (2008)</u>

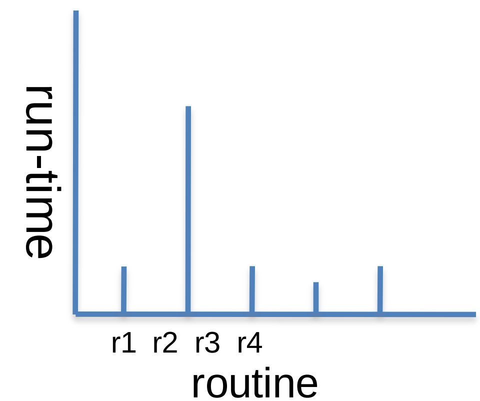
- 16K shared memory
- 16K constant memory
- 2GB global memory

GPU Multi-layer Memory



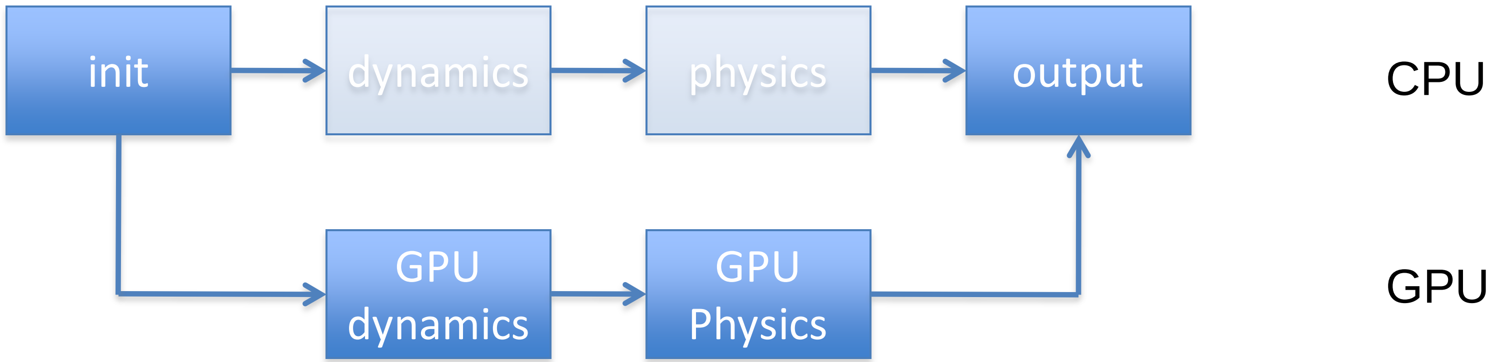
Execution Flow-control
(select routines)





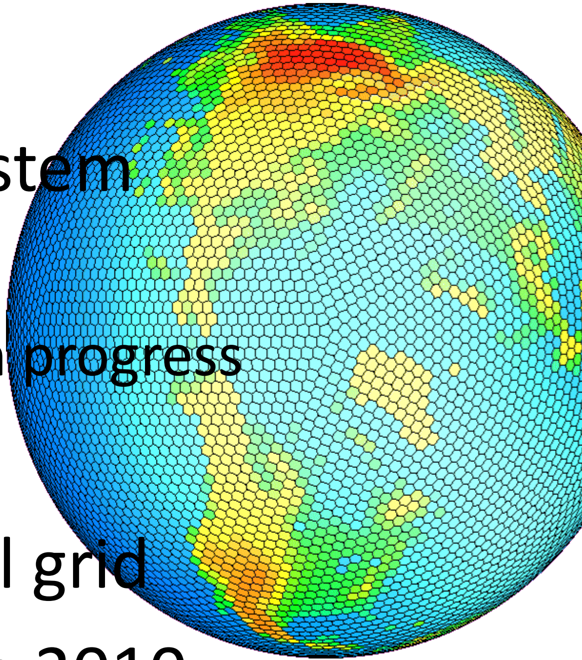
- Copy between CPU and GPU is non-trivial
  - Performance benefits can be overshadowed by the copy
  - WRF demonstrated 20x improvement , 5x overall (Michalakes, 2009)

Execution Flow-control
(run everything on GPUs)



– Eliminates copy every model time step
– CPU-GPU copies only needed for input /output, inter-process communications
– JMA: ASUCA model, demonstrated 70x performance improvement
  - Rewrote the code in CUDA

Non-hydrostatic Icosahedral Model (NIM)

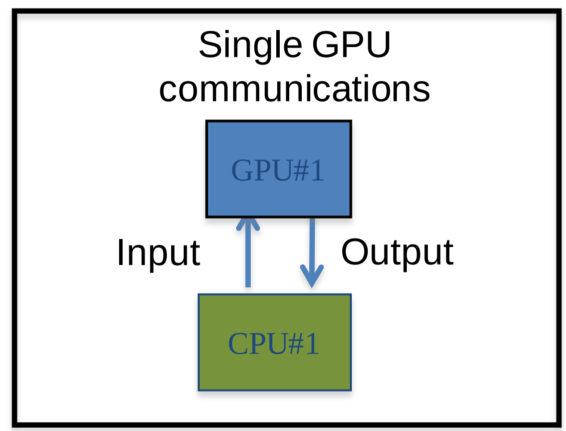


- Global Weather Forecast Model
- Under development at NOAA Earth System Research Laboratory
  - Dynamics complete, physics integration in progress
- Non-hydrostatic
- Uniform, hexagonal-based, icosahedral grid
- Plan to run tests at 3.5km global in late 2010
  - Cloud resolving scale
  - Model validation using AquaPlanet

ESRL
GLOBAL SYSTEMS DIVISION

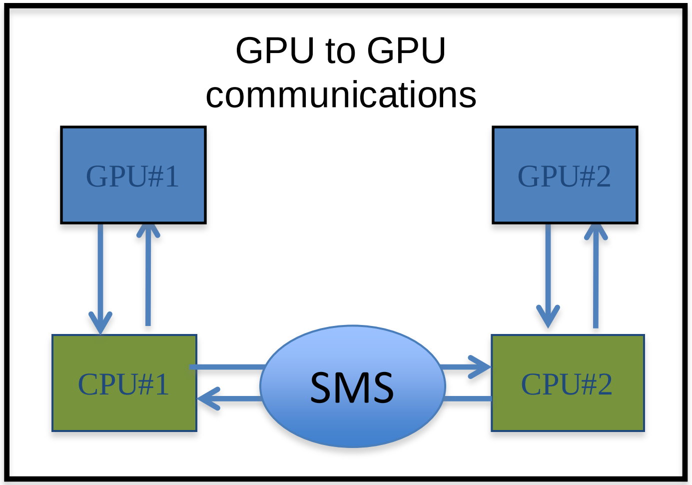
NIM Code Parallelization (2009)

- Developed the Fortran-to-CUDA compiler (F2C-ACC)
  - Commercial compilers were not available in 2008
  - Converts Fortran 90 into C or CUDA-C
  - Some hand tuning was necessary
- Parallelized NIM model dynamics
  - **Demonstrated 34x performance boost over best CPU run time**
    - Tesla Chip, Intel Harpertown (2008)
    - Result for a single GPU
    - Communications only needed for I/O
- Physics parallelization planned



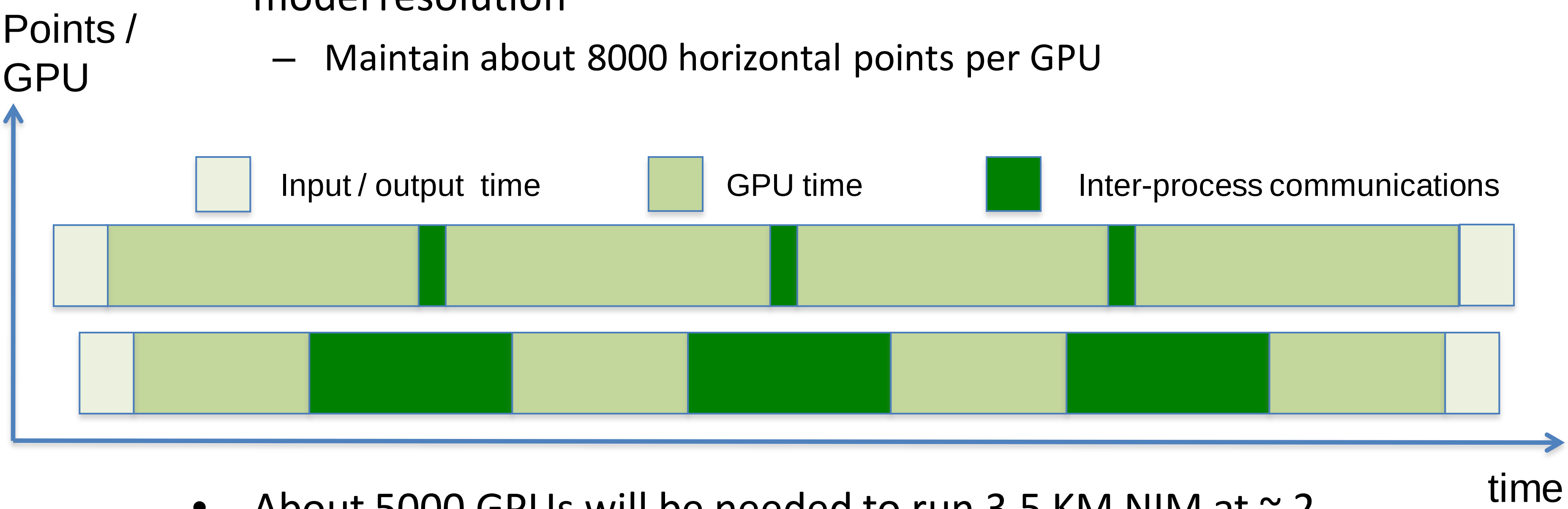
NIM Parallelization Efforts (2010)

- Run on Fermi GPUs
  - ~ 2x improvement over Tesla
- Evaluate Fortran GPU compilers
  - 34x is the benchmark
- Run on Multiple GPUs
  - Modified F2C-ACC GPU compiler
  - Uses MPI-based Scalable Modeling System (SMS)
  - Parallelization is mostly complete



NIM Parallel Performance

- Application scaling is limited by the fraction of time spent doing inter-process communications
  - Minimize data volume and frequency
  - Overlap communications with computations
- Plan to increase the number of GPUs as we increase the model resolution
  - Maintain about 8000 horizontal points per GPU



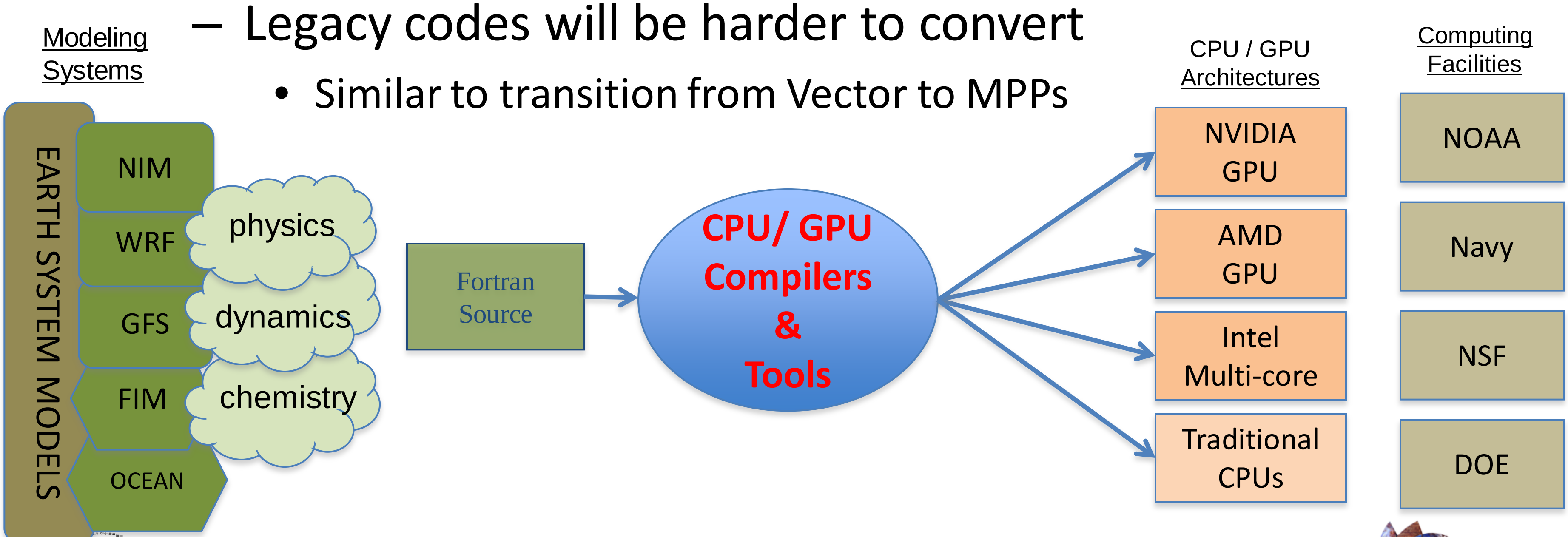
- About 5000 GPUs will be needed to run 3.5 KM NIM at ~ 2 percent of real-time

GPUs and the Challenges Ahead

– Performance and Portability
  - CPUs and GPUs maximize performance differently
  - Challenge to maintain a single source

– New codes are easier to parallelize
  - NIM was designed to run on GPUs
  - Collaboration between model developers, computer scientists

– Legacy codes will be harder to convert
  - Similar to transition from Vector to MPPs



Final Thoughts

- HPC transitions about every decade
  - Vector -> MPP -> COTS Clusters -> GPUs
    - 20x cost savings: hardware, power, infrastructure
- Partnerships
  - Algorithms, tools, compilers, systems, chips
    - Recent DARPA announcement
      - 25 million to advance GPU computing
  - We have had success in GPU computing
    - Compiler development, NIM model parallelization
    - Collaborations with NVIDIA, AMD, PGI, CAPS